VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Detekce vodorovného dopravního značení v obrazech s využitím GPU

Road Markings Detection in Images using GPU

2015

Bc. Jan Štěpán

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jan Štěpán**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Detekce vodorovného dopravního značení v obrazech s využitím GPU
Road Markings Detection in Images using GPU

Zásady pro vypracování:

Inteligentní asisteční systémy v dopravě se stávají důležitým vybavením moderních vozidel. Jedním z takových asistenčních systému je rozpoznávání vodorovného dopravního značení, tj. značení umístěného na vozovce. Cílem této diplomové práce je vytvořit algoritmus pro detekci vodorovného dopravního značení na základě analýzy obrazů získaných kamerou umístěnou v automobilu. Implementaci proveďte s využitím GPU pro urychlení algoritmu. Algoritmus by měl být schopen rozpoznat podélné středové a okrajové čáry a směrové šipky.

Ve své práci proveďte:
1. Popište zadaný problém.
2. Analyzujte řešení a popište potřebnou teorii.
3. Implementujte algoritmus v jazyce C/C++ a proveďte testování v reálných podmínkách.
4. Zhodnoťte dosažené výsledky.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Michael Holuša**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 7. května 2015 ..... Štěpán .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2015 ..... Štěpán .....

Děkuji Ing. Michaeli Holušovi za cenné rady, připomínky a za čas, který mi věnoval při psaní mé diplomové práce. Rovněž chci poděkovat Ing. Janu Gaurovi za zapůjčení grafické karty pro účely testování aplikace.

Abstrakt

Tato diplomová práce si klade za cíl rozpoznávání vodorovného dopravního značení a detekci podélných středových a okrajových čar pruhu, ve kterém se jedoucí vozidlo nachází. Práce je rozdělená do dvou částí, kdy teoretická část popisuje metody použité pro detekcí vodorovného dopravního značení a vlastní implementace pak jednotlivé kroky, které aplikace využívá k detekci a následnému rozpoznání vodorovného dopravního značení na základě pořízeného videozáznamu. Při řešení detekce pruhu, ve kterém se vozidlo nachází, se pro zrychlení zpracování využilo výkonu grafické karty. Dosažené výsledky jsou popsány v závěrečné kapitole, kde je také uvedená procentuální úspěšnost správného rozpoznávání dopravních značek.

Klíčová slova: detekce, OpenCV, CUDA, dopravní značení, prahování obrazu, Houghova transformace, segmentace

Abstract

The goal of the Diploma thesis is the detection of road markings, edge lines, and center lines along the carriageway on which the vehicle is moving. The thesis is divided into two parts. The theoretical part describes the methods used for detection of road markings. The second part, actual implementation, describes the individual steps used to detect and recognize the road markings based on the captured video. The Graphic card (adapter) was used to enhance processing performance of detection of the lines along the carriageway on which the vehicle is located. The results are summarized and described in the final chapter along with the success rate of recognition of the road markings.

Keywords: detection, OpenCV, CUDA, road marking, thresholding, Hough transformation, segmentation

Seznam použitých zkratek a symbolů

BSD licence	–	Berkeley Software Distribution, licence pro svobodný software
CPU	–	Centrální procesorová jednotka(z anglického Central Processing Unit)
CUDA	–	Compute Unified Device Architecture
FPS	–	snímků za sekundu (z anglického Frames per second)
GPU	–	grafický procesor(z anglického Graphic Processing Unit)
HSV	–	Hue, Saturation, Value
MS	–	Microsoft
OpenCV	–	Open Source Computer Vision
RGB	–	Red, Green, Blue
SURF	–	Speeded Up Robust Features
SVM	–	Support Vector Machines

Obsah

1 Úvod **5**

2 Metody digitálního zpracování obrazu použité v diplomové práci **7**
2.1 Barevné modely 7
2.2 Snímaní a digitalizace obrazu 10
2.3 Předzpracování obrazu 11
2.4 Segmentace obrazu 13
2.5 Popis objektu 16
2.6 Příznaky a momenty 16
2.7 Klasifikace 18
2.8 Detekce hran 19
2.9 Houghova transformace 21

3 Vlastní implementace **23**
3.1 Použité technologie 24
3.2 Získání obrazu 24
3.3 Vyhlazení obrazu 25
3.4 Segmentace obrazu 26
3.5 Nalezení objektů 27
3.6 Výpočet vlastností objektu 29
3.7 Porovnání vlastností a určení dopravní značky 31
3.8 Detekce dopravních čar 32
3.9 Vyznačení čar v obraze 38

4 Zhodnocení dosažených výsledků **41**
4.1 Porovnání s bakalářskou prací 44
4.2 Porovnání CPU a GPU verze 45

5 Závěr **47**

6 Reference **48**

Seznam tabulek

1 Vyhodnocení detekce dopravních značek 41
2 Výskyt a detekce jednotlivých dopravních značek 42
3 Konfigurace počítačů, na kterých probíhalo testování 43
4 Vyhodnocení počtu správných určení 44
5 Porovnaní CPU a GPU verze u vybraných operací detekce čar 45
6 Porovnaní CPU a GPU verze výsledných časů zpracovaní snímku 45

Seznam obrázků

1 Aditivní míchání barev 7
2 Grafické znázornění HSV 8
3 Černobílý obraz 10
4 Konvoluce- vytvoření reliéfu 12
5 Prahování 14
6 Segmentace obrazu- prahováním 14
7 Ukázka dilatace 15
8 Ukázka eroze 16
9 Gradient 19
10 Princip Houghovy transformace 21
11 Bodům, které leží na společné přímce, odpovídá v prostoru φ,ρ průsečík sinusoid 21
12 Jednotlivé kroky, prováděné při zpracování obrazu 23
13 Gaussovo vyhlazování maska 25
14 Vyhlazení obrazu 25
15 Porovnání globálního a adaptivního prahování obrazu 26
16 Odstranění malých oblastí z obrazu 27
17 Znázornění možné chybné detekce 28
18 Nalezení a vybraní největšího objektu 29
19 Znázornění konvexní (a) a nekonvexní (b) množiny 30
20 Výsledek po úspěšném rozpoznání dopravní značky 31
21 Vizualizace detekce hran v obraze 33
22 Vizualizace hodnot z akumulátoru 34
23 Vykreslení přímek na základě souřadnic krajních bodů 39
24 Vykreslení detekovaných čar do barevného obrazu 40
25 Ukázka nerozpoznání dopravní značky 42
26 Porovnání časů zpracování jednotlivých operací 46

Seznam výpisů zdrojového kódu

1 Metoda pro detekci hran prováděná na GPU 33
2 Houghova transformace prováděná na GPU 35
3 Nalezení krajních bodu přímky 36
4 Vykreslení čar do obrazu 39

1 Úvod

Žijeme v moderní době, která vyžaduje moderní a dynamicky se rozvíjející technologie. Tyto technologie jsou dnes využívány ve všech oblastech lidské činnosti.

Jak již jsem uvedl žijeme v moderní době, která mimo rozvoje technologii přináší do lidského života stres, uspěchanost a nervozitu. Z tohoto důvodu se domnívám, že je nutné využívat informačních technologií ke zdokonalování a kontrole lidské činnosti, jako je bezpečnost v dopravním provozu či předcházení rizika dopravní nehody.

Dle statistických údajů Policie České republiky [13] bylo v roce 2014 šetřeno 85 859 nehod, při kterých bylo 629 osob usmrceno, těžce zraněno bylo 2 762 osob a 23 655 osob bylo lehce zraněno. Odhadnutá hmotná škoda policií na místě nehody je 4 933,23 mil. Kč. Znamená to, že každých 6 minut a 6 sekund došlo k dopravní nehodě, kterou musela policie prošetřit, každých 13 hodin a 55 minut došlo na našich komunikacích k usmrcení osoby a každých 19 minut a 54 sekund byl zraněn účastník silničního provozu.

Tyto statistické údaje Policie České republiky ukazují, že každý den se na naších silnicích stane v průměru 235,2 dopravních nehod. Velká část z nich je způsobena lidským faktorem. Může to být zaviněno například únavou, zdravotním stavem řidiče, špatným vyhodnocením situace, alkoholem, nepozorností řidiče či používáním mobilních telefonů za jízdy.

Snahou výrobců automobilů je implementace nových technologií, které si kladou za cíl zlepšovat bezpečnost v provozu na pozemních komunikacích. Na trhu jsou dostupné automobily, které využívají lasery, čidla a kamery k zajištění větší bezpečnosti řidiče. Jako příklad lze uvést to, že upozorní řidiče, že nemá zapnutý bezpečnostní pás, nejsou dovřené dveře automobilu, nesvítí světla, dále upozorní na nedostatek tlaku v pneumatikách nebo oleje v motoru, popřípadě na mikrospánek řidiče. Dnes je samozřejmostí, že tato čidla dokážou rozeznat závady na automobilu a řidiče na to upozornit.

Já jsem se rozhodl svou diplomovou prací zaměřit na oblast automobilového průmyslu a zabývat se otázkou detekce vodorovného dopravního značení a rozpoznání o jakou dopravní značku se jedná a to vše v reálném čase.

Definici dopravního značení lze najít v zákoně č.361/2000 Sb., o provozu na pozemních komunikacích, ve znění pozdějších předpisů.

Tento zákon rozlišuje dopravní značky na svislé a vodorovné.

- *Svislé dopravní značky jsou stálé, proměnné a přenosné. Proměnná svislá dopravní značka je dopravní značka, jejíž činná plocha se může měnit. Přenosnou svislou dopravní značkou se rozumí dopravní značka umístěná na červenobíle pruhovaném sloupku (stojánku) nebo na vozidle.*
- *Vodorovné dopravní značky jsou stálé a přechodné. Vodorovné dopravní značky mohou být doplněny dopravními knoflíky.*
- *Tvary symbolů dopravních značek se nesmějí měnit; to neplatí pro dopravní značky se symboly, které mohou být obráceny, a se symboly, které jsou uvedeny jen jako vzory, a pro svislé dopravní značky proměnné.*
- *Prováděcí právní předpis stanoví význam, užití, provedení a tvary dopravních značek a jejich symbolů.*
- *Dopravní značky, světelné a akustické signály, dopravní zařízení a zařízení pro provozní informace musí svými rozměry, barvami a technickými požadavky odpovídat zvláštním technickým předpisům.*

Vodorovné dopravní značky se užívají samostatně nebo ve spojení se svislými dopravními značkami, popřípadě s dopravními zařízeními, jejichž význam zdůrazňují nebo zpřesňují. Vodorovné dopravní značky jsou vyznačeny barvou nebo jiným srozumitelným způsobem; přechodná změna místní úpravy provozu na pozemních komunikacích je vyznačena žlutou nebo oranžovou barvou. [20]

Vycházím ze své bakalářské práce s tím, že nyní použiji jiný způsob vyhodnocení, o kterou dopravní značku se jedná. Ke zrychlení aplikace využiji výkonu grafické karty. Dalším cílem pak bude detekce pruhu, ve kterém se vozidlo nachází.

Diplomová práce bude rozdělená do dvou části. V první části se zaměřím na teorii a vysvětlení pojmů z oblasti zpracování obrazu, které se dají využít pro detekci dopravního značení. Druhá část bude obsahovat popis vlastní implementace daného problému. K otestování použiji videozáznamy, které pořídím na území města Ostravy.

V závěru své práce porovnám výhody a nevýhody zvoleného postupu s výsledky mé bakalářské práce. V té jsem pro rozpoznávání dopravních značek využil principu podobnosti obrázků. Porovnával jsem snímek z videozáznamu se šablonou, která obsahovala dopravní značky, jenž byla aplikace schopna rozpoznat.

2 Metody digitálního zpracování obrazu použité v diplomové práci

Digitální zpracování obrazu patří v současnosti k rychle rozvíjejícímu se odvětví. Při zpracování obrazu se využívají různé barevné modely. Využití a popis některých z nich bude uveden v této kapitole. Dále si představíme jednotlivé kroky, se kterými se můžeme setkat při zpracování obrazu. V první řadě se jedná o snímání a digitalizaci obrazu, dále pak předzpracování a segmentace obrazu, popis objektu až po konečnou klasifikaci objektu.

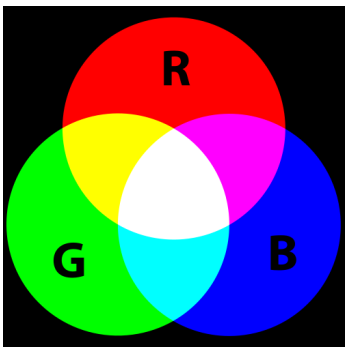
2.1 Barevné modely

V rámci zpracování videozáznamu program využívá k různým operacím následující barevné modely.

2.1.1 Barevný model RGB

RGB model patři mezi nejpoužívanější barevné modely. Jako příklad, kde se s tímto modelem lze, setkat můžeme uvést monitory, projektory a jiná podobná zobrazovací zařízení.

RGB model je tvořen třemi základními barvami tj. červená (Red), zelená (Green) a modrá (Blue). Zbylých barev z barevného spektra lze dosáhnout adaptivním mícháním těchto základních barev. Pokud je intenzita u všech barev nastavena na nulu, je výsledkem černá barva. Bílou barvu lze získat, pokud se všem třem složkám nastaví maximální intenzita. Adaptivní míchání barev je ukázáno na obrázku 1.



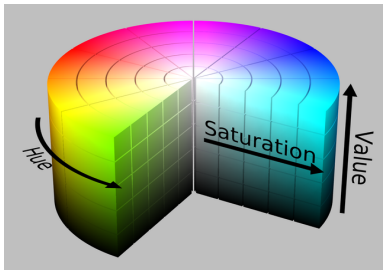
Obrázek 1: Aditivní míchání barev[14]

2.1.2 Barevný model HSV

Tento barevný model nejvíce odpovídá lidskému vnímání barev. Jedná se o barevný model, kde je barva bodu určená třemi složkami. Nejedná se však o tři základní barvy jako tomu je v případě modelu RGB. Tyto složky mají zcela jiný význam.

- **Hue** - Barevný tón neboli odstín. Tato hodnota udává, která barva v daném bodě převládá. Je udávána ve stupních (0 až 360 stupňů).
- **Saturation** - Sytost. Představuje množství šedí vzhledem k odstínu, udává se v % (v rozmezí od 0% pro šedou barvu do 100% pro plně sytou barvu).
- **Value** - Jasová hodnota. Udává světlost nebo tmavost barvy. Vyšší hodnota jasu značí, že barva odráží více světla. Je udávána v %, kde bílá barva má hodnotu jasu 100% a černá barva 0%.

Jak vypadá grafické znázornění modelu HSV je zobrazeno na obrázku 2.



Obrázek 2: Grafické znázornění HSV[7]

Převod RGB modelu na HSV je dán následujícími vzorci:

$$H = \begin{cases} 60° \cdot \left(\frac{G-B}{V-min(R,G,B)}\right) & v = R \\ 60° \cdot \left(\frac{B-R}{V-min(R,G,B)}\right) + 120° & v = G \\ 60° \cdot \left(\frac{R-G}{V-min(R,G,B)}\right) + 240° & v = B \end{cases} \tag{1}$$

$$S = \begin{cases} 0 & v = 0 \\ \frac{V-min(R,G,B)}{V} & v \neq 0 \end{cases} \tag{2}$$

$$V = max\,(R, G, B) \tag{3}$$

Vzorce 1, 2 a 3 popisují převod z barevného modelu RGB na model HSV. Hodnoty R, G, B popisují jednotlivé složky barevného modelu RGB. Jak už je ze vzorců 1, 2 a 3 patrné, převod obrazu z barevného modelu RGB na model HSV, vyžaduje více operací než převod na obraz ve stupních šedi, který je popsán v následující kapitole.

Výhodou barevného modelu HSV je, že umožňuje oddělit informace o barvě od informace o jasové hodnotě daného bodu. Této výhody se využívá při zkoumání barvy jednotlivých bodů, jelikož není ovlivňována jasovou hodnotou. [19]

2.1.3 Obraz ve stupních šedi

Obraz ve stupních šedi, v praxi se lze setkat i s termínem monochromatický, na rozdíl od RGB obrazu, obsahuje pro každý bod pouze jednu hodnotu. Nejčastěji se využívá při tvorbě uměleckých fotografií, při hledání hran v obraze a jejich následném zpracování. Dá se využití i při segmentaci obrazu, kdy lze na základě jasové hodnoty z obrazu vytáhnout jen určité objekty.

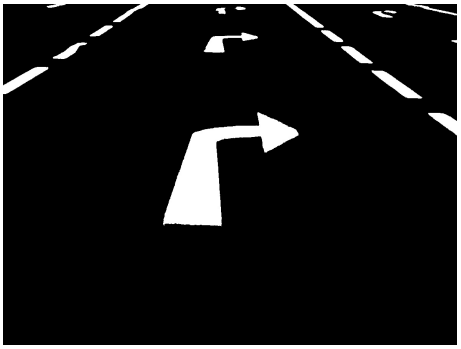
Lidské oko rozlišuje jednotlivé složky RGB s různou citlivostí, proto se nedá využít pouhého sečtení jednotlivých složek, ale pro převod se využívá váhových koeficientů:

$$f = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B, \tag{4}$$

kde f je výsledná úroveň jasu v šedotónovém obraze a hodnoty R, G, B jsou vynásobeny významem úrovní jasu jednotlivých složek daného bodu v barevném obraze.

2.1.4 Binární obraz

Je to digitální jednobitový obraz tvořený pouze dvěma odstíny šedé barvy. Na binárním obraze je typicky každý pixel tvořen černou (označující oblast pozadí) nebo bílou barvou (oblast objektu). Černobílého obrazu lze dosáhnout pomocí prahování obrazu (thresholding). Jak vypadá binární obraz je zobrazeno na obrázku 3.



Obrázek 3: Černobílý obraz

2.1.5 Barevný vs. obraz ve stupních šedi

Při detekci vodorovného dopravního značení je vhodnější použít obraz ve stupních šedi. Při pohledu na vozovku je i lidským okem rozpoznatelné, že barevné spektrum vozovky je ve většině případů tvořeno bílou a šedou barvou. Z tohoto důvodu není nutno při zpracování videa uchovávat informaci o barvě pixelu. Navíc u barevného modelu RGB není jasová hodnota explicitně oddělená od odstínu barvy, a proto je složitější určit intenzitu jasu daného pixelu.

2.2 Snímaní a digitalizace obrazu

Pro zpracování a rozpoznávání obrazu je důležité převést obraz reálného světa do digitální formy. Při snímaní dochází k převodu vstupních veličin na elektrický signál spojitý v čase. Příklady vstupních veličin jsou jas, ultrazvuk, tepelné záření atd. Snímání může probíhat v jednom nebo více spektrálních pásmech. Spektrální pásmo je elektromagnetické záření s vymezeným rozsahem vlnové délky. Pro zachycení barevného obrazu stačí tři spektrální složky (červená, zelená, modrá).

Při digitalizaci je vstupní spojitý signál převeden do diskrétního tvaru, následně je vstupní signál vzorkován a kvantován. Vzorkovaní znamená, že vstupní signál je rozdělen na nezbytné množství malých částí neboli vzorků, které jsou následně

zpracovávány. Tímto způsobem zpracování lze získat konečný počet vzorků. Tyto však stále obsahují velké množství informací.

Vzhledem k této skutečnosti je nutné provést další proces, kterým je tzv. kvantování signálu. Cílem kvantování signálu je zaokrouhlit hodnoty signálu, získaného při vzorkování, a to na předem definované kvantovací úrovně.

Výsledkem digitalizace je matice přirozených čísel, která reprezentuje obraz. Základní a nedělitelnou jednotkou digitálního obrazu je pixel odpovídající jednomu prvku z této matice.[19]

2.3 Předzpracování obrazu

Úkolem předzpracovaní obrazu je potlačení šumu a zkreslení, které vzniklo během digitalizace a přenosu obrazu. Předzpracování obrazu lze využít i při snaze o zvýraznění určitých rysů v obraze, pro jejich následné další zpracovávání.

Mezi základní a často využívané metody předzpracování obrazu patří:

- filtrace a ostření,
- geometrické transformace,
- jasové transformace,
- matematické morfologie.

2.3.1 Konvoluce a vyhlazovaní

Konvoluce se používá pro filtraci obrazu jako je zvýraznění hran nebo vyhlazení obrazu. Při konvoluci se používá tzv. konvoluční maska. Jedná se o matematickou operaci s maticí, která kombinuje dvě vstupní matice za účelem získaní matice třetí. První matice udává použitý obraz, který je určený k úpravě. Druhá matice (tzv. maska) udává jaký efekt s obrazem má být proveden. Výsledek této operace je následně zapsán do třetí matice.

V grafických editorech se nejčastěji používá maska o velikosti 3×3 nebo 5×5. Tyto velikosti jsou dostatečné pro většinu požadovaných efektů.

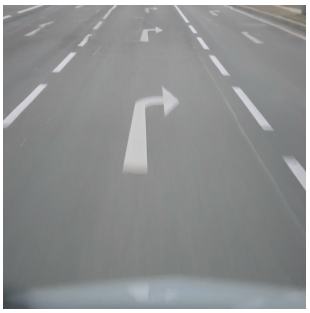
Všechny pixely obrazu jsou postupně zpracovávány tímto filtrem. Hodnota aktuálního pixelu středu masky je vynásobená hodnotami osmi sousedních pixelů.

Výsledné hodnoty jsou poté sečteny a přiřazeny aktuálnímu pixelu dle vzorce:

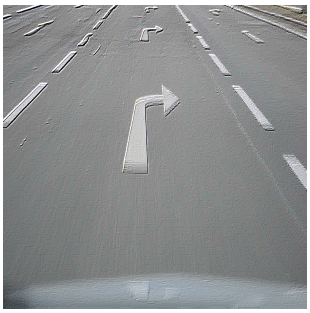
$$g(x,y) = h \cdot f = \sum_{s=-s_{\max}}^{s_{\max}} \sum_{t=-t_{\max}}^{t_{\max}} h(s,t)\, f(x-s, y-t), \tag{5}$$

kde s_{max} a t_{max} vyjadřují polovinu rozměrů masky, $f(x,y)$ je vstupní obraz a $h(s,t)$ je tzv. jádro. Výsledný obraz je pak označen $g(x,y)$.

Pomocí konvoluce lze obrázek rozmazat, zaostřit, detekovat hrany nebo vytvořit reliéf. Jak vypadá vytvoření reliéfu je uvedeno na obrázku 4. Na obrázku 4(a) je zobrazen původní snímek, obrázek 4(b) ukazuje, jak vypadá použité jádro, pomocí kterého se dosáhne efektu reliéfu. Výsledek tohoto efektu je znázorněn na obrázku 4(c).[19] Pro lepší znázornění ve výsledném obrázku byla tato operace použita dvakrát za sebou.



-2	-1	0
-1	1	1
0	1	2



(a) (b) (c)

Obrázek 4: Konvoluce- vytvoření reliéfu[4]

2.4 Segmentace obrazu

Jedním z nejdůležitějších kroků vedoucích k analýze obsahu na obrázku je segmentace obrazu. Jeho snahou je rozčlenit obraz na části, které nějak souvisí s předměty či oblastmi (objekty) reálného světa a pozadí.

Segmentaci dělíme na kompletní nebo částečnou. U kompletní segmentace má být výsledkem soubor vzájemně nepřekrývajících se oblastí, které jednoznačně korespondují s objekty vstupního obrazu. V případě, kdy vytvořené segmenty přímo nesouhlasí s objekty obrazu, jde o segmentaci částečnou.

Za výhodu segmentace obrazu lze považovat výrazné snížení objemu zpracovaných dat. Mezi problémy, ovlivňující segmentaci, patří nejednoznačnost obrázkových dat, která je často doprovázená informačním šumem.[6]

Pro názornost lze uvést některé ze segmentačních metod: segmentace narůstáním oblastí, segmentace srovnáním se vzorem, segmentace prahováním, segmentace na základě detekce hran.

2.4.1 Segmentace prahováním

Jednou z nejstarších a nejjednodušších metod pro segmentaci obrazu je prahování. Zároveň je také, díky nízké výpočetní náročnosti, nejrychlejší segmentační metodou a je tedy vhodná pro zpracování obrazu v reálném čase. Pomocí zvolené jasové konstanty neboli prahu lze oddělit objekty od pozadí.

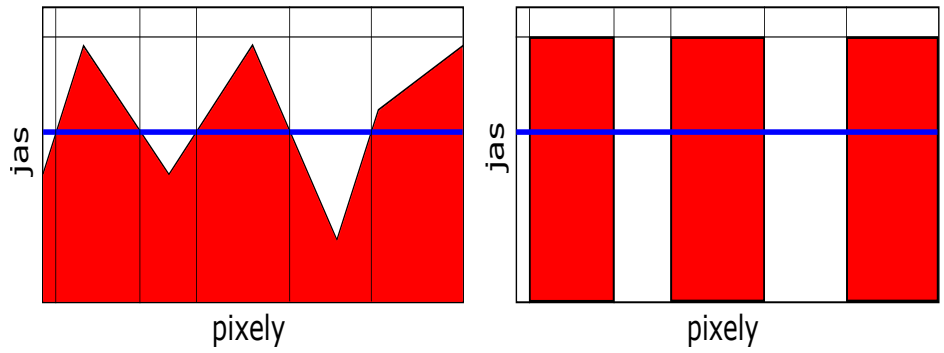
Prahování je transformace vstupního obrazu f na výstupní (segmentovaný) binární obraz g podle vztahu:

$$g(x,y) = \begin{cases} 1 & \text{pro } f(x,y) \geq T, \\ 0 & \text{pro } f(x,y) < T \end{cases} \tag{6}$$

Pro vzorec 6 platí, že T je předem zvolená konstanta nazvaná práh, $g(x,y) = 1$ pro obrazové elementy náležející po segmentaci objektům a $g(x,y) = 0$ pro elementy pozadí (nebo naopak).

Na obrázku 5(a) je ukázaná intenzita jednotlivých pixelů. Modrá čára pak reprezentuje zvolenou hodnotu prahu.

Metoda segmentace prahováním se používá pro převedení obrazu, který obsahuje více úrovní jasu (odstínů šedi) na obraz, ve kterém se vyskytují dvě jasové úrovně (černá a bílá). Pokud je hodnota jasu v daném bodě větší než práh (threshold), je danému bodu přiřazena hodnota 1 (případně 255).

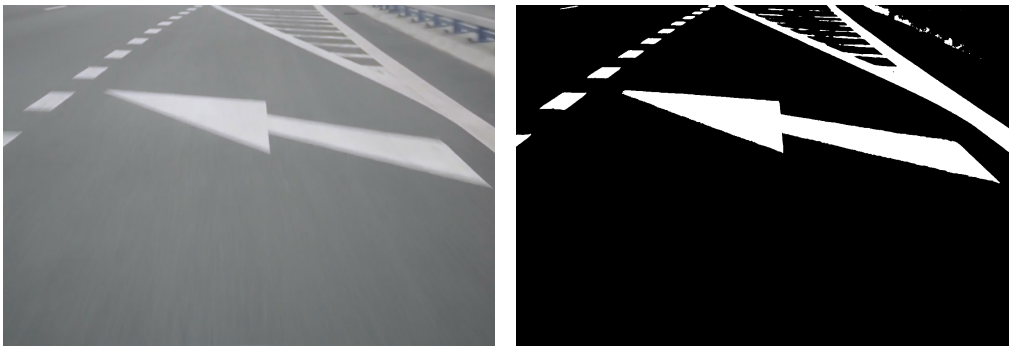


(a) Hodnoty jasu v původním obrázku (b) Hodnoty jasu po binárním prahováním

Obrázek 5: Prahování

Prahování může být globální (statické) nebo lokální (adaptivní). U globálního prahování je zvolená hodnota prahu, která platí pro celý obraz. Vhodné použití tohoto typu prahování je u obrazů, kde je neměnný jas, jako jsou např. psaný text, krevní buňky apod. Globální práh je určován z celého obrazu.

Jak vypadá segmentace obrazu prahováním je zobrazeno na obrázku 6. Na obrázku 6(a) je znázorněn původní obrázek a na obrázku 6(b) je výsledek po prahování. Na výsledném obrázku je vidět, že obsahuje pouze dvě barvy a to bílou a černou. Bílá barva je přiřazena oblastem, které mají hodnotu jasu vyšší než je zvolený práh, zbylým oblastem je přiřazena černa barva.



(a) (b)

Obrázek 6: Segmentace obrazu- prahováním

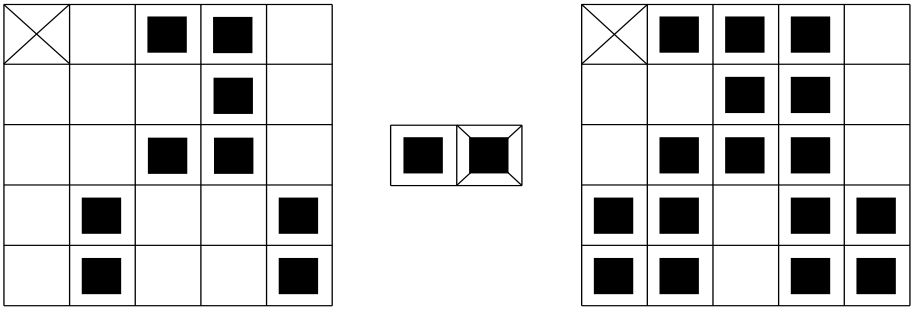
2.4.2 Matematická morfologie

Operace matematické morfologie se využívají zejména při zpracování binárního obrazu, který se získá segmentací obrazu.

Za základní operace matematické morfologie se považují eroze a dilatace. Tyto operace lze využít během předzpracování obrazu pro odstranění šumu nebo zjednodušení tvaru objektů. Rovněž je lze použít pro zvýraznění struktury objektů (ztenčovaní nebo zesílení objektů). Při provádění operací dilatace a eroze se podobně jako u konvoluce využívá tzv. masky. Tato maska nám popisuje co se provede s daným pixelem na základě okolních bodů.

Dilatace znamená, že se hranice daného objektu rozšíří o jeden pixel na úkor pozadí. Díky této operaci tak můžeme odstranit malé díry, které ležely uvnitř objektu.

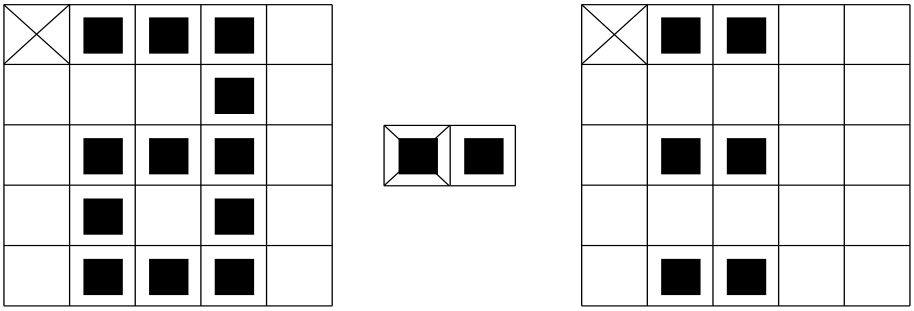
Na obrázku 7 je ukázka provedení dilatace. Z masky vyplývá, že pokud je pixel bílý, pak se pixel nalevo od něj vykreslí bílou barvou.



Obrázek 7: Ukázka dilatace

Eroze se využívá pro zmenšení daného objektu nebo pro odstranění malých objektů. Při provádění eroze se hranice objektu zmenší o jeden pixel. Ukázka provedení eroze je na obrázku 8. Maska nám říká, že pokud napravo od pixelu leží bílý pixel, tak se daný pixel zachová. Pokud je tomu jinak, pixel se odstraní.

Tyto metody jsou základem dalších operací matematické morfologie jako jsou otevření nebo uzavření. Otevření je postupné provedení eroze a následné dilatace. Uzavření je opakem otevření, tedy nejprve se provede operace dilatace následovaná erozí.



Obrázek 8: Ukázka eroze

2.5 Popis objektu

Po provedení segmentace obrazu přichází na řadu popis nalezených objektů v obraze. To lze provést dvěma způsoby, a to kvantitativně (pomocí číselných charakteristik), nebo kvalitativně (pomocí relací mezi objekty). Zvolený způsob, jak se daný objekt popíše, se odvíjí od toho k čemu se popis bude používat. Jedním z nejčastěji používaných způsobů popisů je popis na základě velikosti objektu neboli počet pixelů, které danému objektu v obraze odpovídají.

2.6 Příznaky a momenty

V této kapitole jsou ukázány vzorce pro výpočet momentů a také příklady příznaků.

2.6.1 Momenty

Díky schopnosti rozlišit od sebe třídy charakterizující různé objekty a pro jejich relativně jednoduchý výpočet patří momenty různých stupňů k často používaným příznakům. Momentů se pak využívá například pro výpočet těžiště objektu.

Moment, který se vztahuje k souřadné soustavě obrazu je definován následujícím vztahem:

$$m_{p,q} = \iint\limits_{\Omega} x^p y^q f\left(x,y\right) dxdy, \tag{7}$$

kde $f\left(x,y\right)$ je obrazová funkce, p respektive q nám udávají stupeň momentu a Ω představuje část obrazu, kterou považujeme za rozpoznávaný objekt. Vztah 7 odpovídá spojité obrazové funkci. Pro diskrétní obrazovou funkci by stačilo integraci nahradit sumací. Hodnoty p a q se volí pod podmínkou $p \geq 0$ a $q \geq 0$.

Je-li pro řešení nějakého problému důležitá plocha objektu, využívá se momentu $m_{0,0}$, který odpovídá ploše objektu váženou hodnotou jasu.

Někdy je vhodné, pro rozhodnutí o jaký objekt se jedná, zvolit jako příznak těžiště objektu. K výpočtům souřadnic těžiště objektu lze využít momentů $m_{0,0}$, $m_{1,0}$ a $m_{0,1}$.

$$x_t = \frac{m_{1,0}}{m_{0,0}}, y_t = \frac{m_{0,1}}{m_{0,0}}, \tag{8}$$

kde x_t respektive y_t označují souřadnice těžiště na osách x a y.

2.6.2 Příznaky

Příznaky jsou vlastnosti, na základě kterých můžeme od sebe rozlišit různé typy objektů jako jsou čtverec, obdélník, kruh nebo hvězdy. Pro rozhodnutí, do které z těchto skupin objekt patří, lze využít následujících vlastností (příznaků):

- pravoúhlost,
- podlouhlost,
- kruhovost,
- energie hranice,
- průměrná vzdálenost pixelu od hranice.

Pravoúhlost

Při výpočtu pravoúhlostí rotujeme postupně hranicí objektu v rozmezí $0 - 90°$. Dále je vhodné zvolit nějaký krok úhlu rotace, o který budeme daným objektem rotovat (např. $5°$). Po provedení rotace se okolo hranice objektu opíše pravoúhelník, který bude mít strany vždy rovnoběžné se stranami obrazu. Nakonec se porovnají velikosti všech těchto pravoúhelníků a vybere se ten s nejmenší plochou.

$$R = \frac{A_O}{A_R} \tag{9}$$

Podlouhlost nám určuje poměr stran daného minimálního pravoúhelníku.

$$S = \frac{a}{b}, \tag{10}$$

kde a a b označují strany nejmenšího pravoúhelníku.

Kruhovost objektu je definovaná vztahem:

$$C = \frac{P^2}{A}, \tag{11}$$

kde P označuje délku hranice objektu a A jeho plochu.

Např. pro čtverec vychází hodnota kruhovosti $C = 16$, pro kruh $C = 4\pi$ a pro objekty, které mají nepravidelný tvar vycházejí hodnoty vyšší. [17]

2.7 Klasifikace

Poslední krok, který se využívá při zpracování obrazu, je klasifikace objektů. Cílem klasifikace objektů je roztřídit nalezené objekty do skupin předem známých tříd. Jako příklad takovýchto tříd lze uvést kulaté či hranaté objekty. Při klasifikaci se využívá dvou základních skupin metod. Jedná se o metodu příznakového rozpoznávání a metodu strukturálního rozpoznávání.

Příznakové metody rozpoznávání, jak už ze samotného názvu metody vyplývá, využívají principu tzv. příznaků objektů. Nejvhodnějším využitím této metody je vybrat minimální množství příznaků, které dané objekty co nejlépe popíšou. Při využití více příznaků získáme tzv. vektor příznaků. Ten nese veškeré podstatné informace o daném objektu. Při následném rozpoznávání se využívá pouze těchto vektorů. Jako příklad příznaků lze uvést pravoúhlost nebo kulatost.

Strukturální metody rozpoznávání fungují na principu, kdy obraz je složen ze základních popisných elementů (tzv. primitiv), jejich vlastností a vztahů mezi nimi.

V praxi lze využít obou metod, tedy jak příznakové tak strukturální, současně, jelikož se vzájemně doplňují.[19]

2.7.1 Klasifikační metody

Pro klasifikaci objektů lze využít několika známých metod. Tyto metody lze rozdělit do dvou základních skupin. Jedná se o klasifikační metody s učením a klasifikační metody bez učení.

Klasifikační metody s učením

Metody, které patří do této skupiny, využívají ke svému učení trénovací množinu dat. Na základě zpětné vazby jsou tyto metody schopné zdokonalovat své rozhodovací kritéria. Jako příklad často využívané metody s učením lze uvést Neuronovou síť, která se snaží napodobit fungování lidské nervové soustavy.

Další metodou, jenž využívá učení na trénovací množině je SVM. SVM při klasifikaci hledá nadrovinu, pomocí které se v oblasti s příznaky optimálně rozdělí trénovací data.

Klasifikační metody bez učení

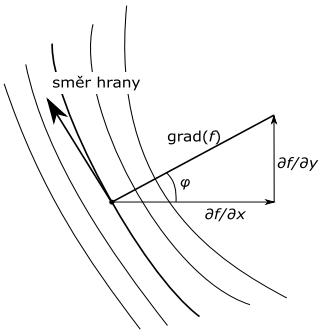
Mezi metody, které ke klasifikaci objektů nevyužívají učení na trénovací množině patří metody využívající podobnosti dvou obrázků. V OpenCV je to např. metoda MatchTemplate, kde se jeden obrázek postupně posouvá po druhém a hledá se místo, ve kterém je dosažená určitá podobnost. Pokud je tato podobnost (určená v procentech) vyšší než předem stanovená hodnota, lze říci že objekty patří do stejné třídy.

2.8 Detekce hran

Hrany v obraze odpovídají prudkým změnám hodnot jasu. Základní myšlenkou detekce hran je najít místa v daném obraze, kde dochází k významné změně jasu. Tyto jasové změny lze detekovat pomocí prvních a druhých derivací intenzity jasu. Základním kritériem pro nalezení takovýchto změn je výše hodnoty první a druhé derivace intenzity jasu případně změna znaménka derivace. [2]

2.8.1 První derivace, gradient

Pro rozhodnutí, zda daný bod je nebo není součástí hrany, je rozhodující směr ve kterém je změna jasu největší. Tímto směrem je gradient obrazové funkce. Směr hrany je kolmý ke směru gradientu. Velikost hrany se rovná velikosti gradientu.



Obrázek 9: Stanovení směru a velikosti hrany na základě gradientu[17]

Označme velikost hrany jako $e(x, y)$, směr gradientu jako $\varphi(x, y)$ a směr hrany v bodě (x, y) jako $\psi(x, y)$. Dále zaveďme značení $f_x(x, y) = \partial f(x, y)/\partial x$ a $f_y(x, y) = \partial f(x, y)/\partial y$.

Pak platí:

$$e(x, y) = \sqrt{{f_x}^2(x, y) + {f_y}^2(x, y)}, \tag{12}$$

$$\varphi(x, y) = \arctan\left[\frac{{f_y}^2(x, y)}{{f_x}^2(x, y)}\right], \tag{13}$$

$$\psi(x, y) = \varphi(x, y) + \frac{\pi}{2}. \tag{14}$$

Cílem tohoto výpočtu je rozhodnout zda bod na souřadnicích x a y je nebo není součásti hrany (hranice) nějakého objektu. V jednoduchých případech lze za hraniční body považovat ty, kde hodnota $e(x, y)$ je vyšší než předem zvolená prahová hodnota.[17]

Existuje několik metod, které pro nalezení hrany využívají výpočtu velikosti gradientu. Patří mezi ně například Robertsův operátor, operátor Prewittové, Sobelův operátor nebo Cannyho detektor hran. Definici a vzorce pro výpočet velikosti hrany pomocí těchto operátorů jsou uvedeny v [17](kapitola 8.1.1 Gradientní metody hledaní hran).

2.9 Houghova transformace

Za předpokladu, že jsme použili některou z metod pro detekci bodů ležících na hraně objektu, je dalším krokem určit zda tyto hraniční body lze proložit přímkou. Metoda, která se využívá k nalezení této přímky, se nazývá Houghova transformace.

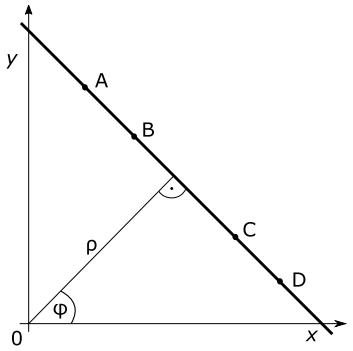
Přímku v rovině (x, y) lze vyjádřit následujícím vzorcem:

$$\rho = x \cos \varphi + y \sin \varphi, \tag{15}$$

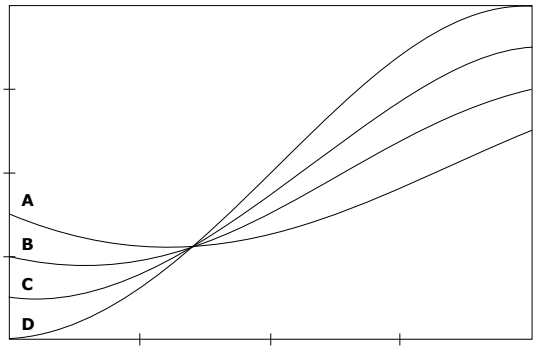
kde ρ je kolmá vzdálenost od počátku souřadného systémů a φ označuje úhel.

Máme-li bod H, který leží na souřadících (x_H, y_H), pak pro přímku která prochází tímto bodem platí: $\rho = x_H \cos \varphi + y_H \sin \varphi$. Uvažujme nyní prostor φ, ρ. Pak v tomto prostoru odpovídá každé jednotlivé přímce, která prochází daným bodem H právě jediný bod. Svazku přímek jenž procházejí bodem H v tomto prostoru odpovídá sinusoida $\rho = x_H \cos \varphi + y_H \sin \varphi$.

Nyní si představme, že body A, B, C, D leží na společné přímce (obrázek 10), pak v prostoru φ,ρ odpovídají svazkům možných přímek sinusoidy A, B, C, D (obrázek 11). Obrazu přímky, která prochází všemi danými body A, B, C, D odpovídá v prostoru φ,ρ bod, ve kterém se všechny sinusoidy A, B, C, D protínají.



Obrázek 10: Princip Houghovy transformace



Obrázek 11: Bodům, které leží na společné přímce, odpovídá v prostoru φ,ρ průsečík sinusoid

Rozdělme prostor φ,ρ na oblasti, kde dvojice (i, j) bude označovat obdélníkovou oblast pro kterou platí: $\varphi_{i-1} \leq \varphi < \varphi_i$, $\rho_{j-1} \leq \rho < \rho_j$. Nyní si nad tímto prostorem φ, ρ zaveďme dvourozměrný akumulátor $h\,(i, j)$. Nejprve je potřeba všechny hodnoty v akumulátoru nastavit na hodnotu 0. Následně se projde celý obraz a pokud se v daném pixelu $H\,(x_H, y_H)$ nachází bod hrany, sestrojíme v prostoru φ, ρ sinusoidu. Zároveň ve všech bodech $h\,(i, j)$, kterými daná sinusoida prochází, inkrementujeme hodnotu v akumulátoru. Poté, co takto projdeme celý obraz, získáme počet bodů (n) které leží na dané přímce $h\,(i, j) = n$, jenž je charakterizována hodnotami $\varphi \in \langle \varphi_{i-1}, \varphi_i)$, $\rho \in \langle \rho_{j-1}, \rho_j)$.[17]

Tohoto akumulátoru se využije k získání jen těch přímek, které mají délku větší nebo rovnu předem zvolené prahové hodnotě.

Pomocí následujících vzorců, které jsou odvozené ze vzorce 15, můžeme pro tyto přímky vypočítat body, kterými přímky prochází na ose x a y a to jednoduše tím, že do rovnice dosadíme $y = 0$ pro výpočet x respektive $x = 0$ pro y.

$$y = \frac{r - x\cos\varphi}{\sin\varphi}, \quad x = \frac{r - y\sin\varphi}{\cos\varphi}, \tag{16}$$

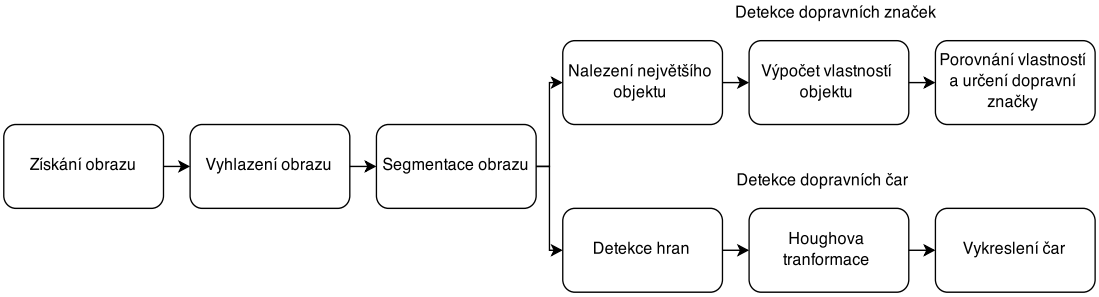
3 Vlastní implementace

Tato v pořadí třetí kapitola se zaměřuje na vlastní implementaci daného problému. Budou zde popsány jednotlivé kroky počínaje získáním videozáznamu z kamery, předzpracováním jednotlivých snímků, následované segmentací obrazu pro získání objektů zájmu. Kapitola se bude rovněž zabývat metodami pro odstranění šumu a malých objektů. Následně se pro objekty, které mohou být považovány za dopravní značku vypočtou její momenty a ty se posléze porovnají s momenty jednotlivých dopravních značek. Rovněž zde bude popsán způsob řešení detekce pruhů.

V této diplomové práci aplikace rozpoznává následující dopravní značky:

- šipka rovně,
- šipka doleva nebo doprava,
- šipka rovně a doleva nebo rovně a doprava,
- předběžné šipky doleva nebo doprava.

Jednotlivé kroky zpracování snímku jsou znázorněný na následujícím obrázku.



Obrázek 12: Jednotlivé kroky, prováděné při zpracování obrazu

3.1 Použité technologie

V rámci aplikace na detekci vodorovného dopravního značení byla využita knihovna OpenCV a technologie CUDA. Technologie CUDA byla aplikována z důvodu zrychlení algoritmu pomocí paralelního zpracování na grafické kartě. Knihovna OpenCV nabízí relativně jednoduchou implementaci a možnost zpracovaní a manipulaci s obrazem.

OpenCV

Jedná se o multiplatformní knihovnu obsahující funkce pro zpracování a manipulaci s obrazem v reálném čase. Je uvolněná pod BSD licencí a její využití je tedy zdarma nejen pro akademické, ale i pro komerční účely. S jejím využitím je možné se setkat při analýze a zpracování snímků z hornictví, lékařství, interaktivního umění nebo u robotů. Aplikace, jenž jsou napsané a optimalizované v C/C++, pak mohou pro svůj běh využívat vícejádrové procesory.

CUDA

Jedná se o platformu vytvořenou firmou NVIDIA pro paralelní programování a výpočty. Je to hardwarová a softwarová architektura, která umožňuje spouštět programy napsané v jazycích C/C++, FORTRAN na GPU (grafickém procesoru). Využití teto architektury je však možné pouze na grafických kartách společnosti NVIDIA.

3.2 Získání obrazu

Při přehrávaní videosekvence dochází ke zpracování každého snímku samostatně. Aplikace je schopna zpracovat snímek v libovolném rozlišení, ale pro zajištění dostatečné rychlostní zpracovaní každého snímku je video na vstupu zmenšeno na rozlišení 640×480 pixelů. Díky tomu je aplikace schopna zpracovávat video o frekvenci 25 FPS v reálném čase. V dalším kroku dochází k úpravě snímku za účelem zmenšení oblasti nebe a odstranění kapoty automobilu. Pro následné zpracování se snímek převádí na monochromatický obraz.

3.3 Vyhlazení obrazu

Vyhlazení obrazu je jednoduchá a často používaná operace, která slouží k odstranění nežádoucího šumu z obrazu. Často používanými metodami pro vyhlazení obrazu jsou Gaussovo vyhlazování nebo vyhlazování průměrováním.

V této diplomové práci byla použita metoda Gaussova vyhlazování. Tato metoda funguje na principu že se koeficientům, které se nacházejí blíže ke středu masky, přiřadí vyšší hodnoty, což odpovídá hodnotám na Gaussově křivce. Maska je daná dvěma vstupními parametry. Jedná se o dvě kladná lichá čísla. Díky této masce se určí oblast, ze které se bude vypočítávat hodnota jasu pixelu ležícího ve středu této oblasti.

Jak vypadá taková maska je ukázáno na obrázku 13. Obrázek 14 ukazuje jak bude vypadat výsledek 14(b) pro původní snímek ve stupních šedi 14(a) při použití této masky.

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Obrázek 13: Gaussovo vyhlazování maska

(a) (b)

Obrázek 14: Vyhlazení obrazu

3.4 Segmentace obrazu

Při detekci vodorovného dopravního značení a detekci čar označujících pruhy se dá využít faktu, že toto značení se v České republice standardně provádí bílou barvou, případně žlutou barvou pro přechodné dopravní značení. Díky tomuto standardu lze v obraze hledat pouze oblasti, které mají bílou nebo žlutou barvu.

K nalezení takovýchto oblastí se dá využít některá z metod pro prahování obrazu, které jsou popsány v kapitole 2.4.1. Aby bylo možné využít metody pro prahování obrazu za účelem nalezení oblastí zájmů, je nutné nejprve vstupní barevný obraz, tvořený třemi složkami (red, green, blue) převést na obraz ve stupních šedí. Tento převod se provede pomocí metody na konvertování barev, která převede vstupní obraz na základě vzorce 4 na výstupní monochromatický obraz (ten je charakteristický tím, že je tvořen pouze jednou složkou a tou je jas).

Pro segmentaci je možné využít dvou typu prahování a to globálního nebo lokálního, nazývaného také adaptivní prahování. Rozdíl mezí jednotlivými typy prahování, jak už z jejich názvů vyplývá, je oblast ze které se vypočítavá prahovací hodnota. V této aplikaci byly nejprve otestovány oba tyto typy. I přes skutečnost, že pomocí adaptivní metody lze dosáhnout lepších výsledků (je schopna lepé reagovat na časté změny jasu) bylo nakonec vybráno globální prahování a to vše z důvodu rychlosti zpracovaní jednoho snímku.

Na obrázku 15 se vyobrazeno porovnání globálního a adaptivního prahování. Pro globální prahování byla prahová hodnota nastavena na 175. U adaptivního prahování se hodnota pixelu vypočítávala z oblasti o velikosti 115×115, kdy daný pixel ležel uprostřed této oblasti. Hodnota pak byla vynásobená váhovou hodnotou -2. Tyto hodnoty byly vybrány na základě testování.

Jak vypadají výsledky obou metod je zobrazeno na obrázku 15(b) a obrázku 15(c).

(a) Snímek vybledlé dopravní značky (b) Globalní prah (c) Adaptivní prahování

Obrázek 15: Porovnání globálního a adaptivního prahování obrazu

3.5 Nalezení objektů

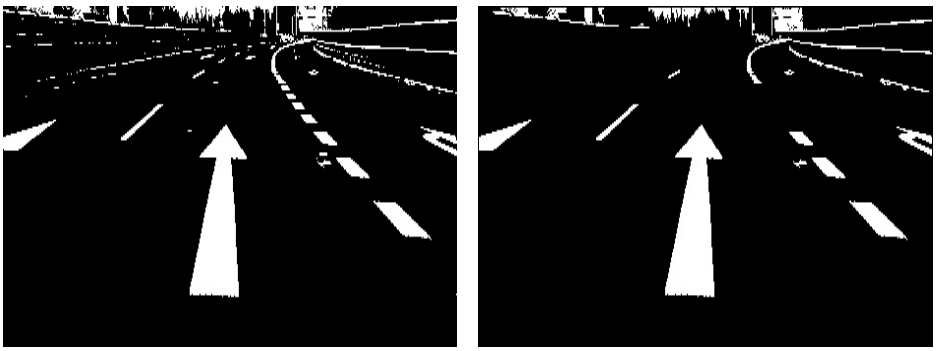
S obrazem, získaným po segmentaci (obrázek 16(a)) se provede operace, která na daném obraze vyhledá všechny bílé objekty. Objekty zaindexuje a uchová si ke každému souřadnice, na kterých se daný objekt nachází a to včetně jeho rozměrů. Díky této operaci je možné vybrat jen ty objekty, které mohou být považovány za dopravní značku nebo označení pruhu.

V této diplomové práci byla pro tyto účely využita metoda FindContours, která se nachází v knihovně OpenCV. Tato metoda pro své použití vyžaduje binární obraz. Metoda projde tento binární obraz a všechny nalezené kontury si uloží do stromové struktury. Ke každé kontuře jsou uloženy sekvence bodů, které popisují obrys objektu.

3.5.1 Odstranění malých oblastí objektů

Je možné použít několik způsobu pro odstranění malých oblastí objektů popřípadě šumu. Patří mezi ně například matematická morfologie. Do této kategorie spadá operace zvaná eroze, která je popsána v kapitole 2.4.2. Využití této operace v tomto případě není úplně vhodné a to z důvodu možného porušení původního tvaru dopravní značky, což může vést k následnému chybnému rozpoznání této dopravní značky.

Druhou variantou, jak odstranit malé nežádoucí oblasti z obrazu, je odstranění těchto oblastí na základě velikosti respektive obsahu této oblasti. Této varianty je možné využít díky faktu, že vodorovná dopravní značka bude v obraze zabírat relativně velkou část a tím pádem bude mít velký obsah.



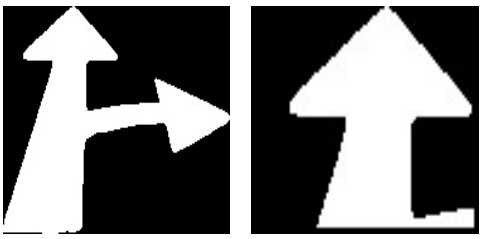
(a) (b)

Obrázek 16: Odstranění malých oblastí z obrazu

Jak vypadá takovýto obraz, po odstranění malých objektů, je ukázáno na obrázku 16. Z obrázku je patrné, že odstraněním těchto oblastí může dojít i k odstranění vzdálených částí přerušované čáry.

3.5.2 Nalezení největšího objektu

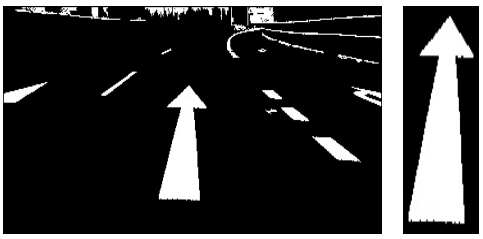
Stejného principu, kterého se využívá pro odstranění malých objektů, se dá využít i pro nalezení toho největšího. Aplikace je napsána tak, aby v obraze detekovala a následně rozpoznávala pouze ty dopravní značky daného dopravního pruhu, ve kterém se řidič s vozidlem nachází. Kamera je z automobilu namířená na střed tohoto pruhu a proto lze hledat objekty, které mohou být považovány za dopravní značku, v prostřední třetině obrazu. Abychom se vyvarovali špatné detekce dopravní značky, např. značka je částečně zakrytá automobilem (a to jak naším nebo jedoucím před námi), je nutné obraz oříznout z horní a dolní strany. Tato situace je znázorněna na obrázku 17, kdy v určitém okamžiku už odbočení doprava není detekovatelné.



(a) (b)

Obrázek 17: Znázornění možné chybné detekce

Po nalezení největšího objektu v obraze je pro tento objekt vypočten minimální obdélník, který ho opíše. Na základě souřadnic objektu a rozměrů tohoto obdélníku se z obrazu vyřízne pouze tato oblast. Znázornění této operace je ukázáno na obrázku 18. Obrazu 18(b), který se získal po nalezení největšího objektu, se využívá pro následné vypočtení vlastností a momentů daného objektu.



(a) (b)

Obrázek 18: Nalezení a vybraní největšího objektu

3.6 Výpočet vlastností objektu

Pro rozpoznávaní objektů je potřeba vytvořit určité příznaky (vlastnosti), které nám budou charakterizovat a vzájemně od sebe odlišovat jednotlivé skupiny (typy) objektů (popsáno v kapitole 2.7). Pro využití některých z těchto vlastností je potřeba vypočítat minimální opsaný obdélník objektu, který bude mít rozměry minimální šířky a výšky daného objektu. V této diplomové práci byly jako příznaky zvoleny následující vlastnosti:

- poměr obvodu a obsahu objektu

$$C = \frac{O}{N}, \tag{17}$$

  kde O znamená obvod objektu a N plochu objektu neboli počet pixelu daného objektu.

- podlouhlost objektu

$$L = \frac{a}{b}, \tag{18}$$

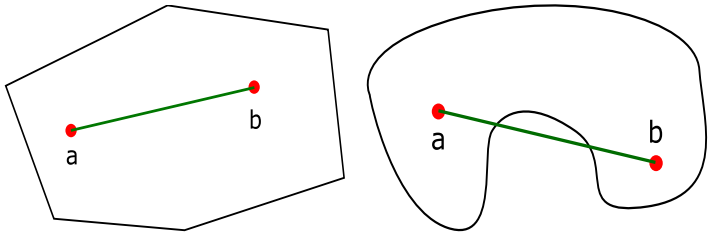
  kde a značí delší a b kratší stranu minimálního opsaného obdélníku.

- pravoúhlost objektu

$$R = \frac{N}{S}, \tag{19}$$

  kde $S = a \times b$ znamená obsah plochy nejmenšího opsaného obdélníku a N plochu objektu neboli počet pixelu daného objektu.

- tzv. jednolitost, jenž je daná jako poměr obsahu oblasti objektu a konvexního obalu objektu [18].



Obrázek 19: Znázornění konvexní (a) a nekonvexní (b) množiny

Pro rozhodnutí o orientaci dopravní značky (šipky) byly použity momenty $m_{0,0}$, $m_{1,0}$ a $m_{0,1}$ pomocí kterých se za využití vzorce 8 provedl výpočet těžiště objektu. Následně jsou vypočteny centrální momenty (μ_{ij}) a ty jsou pomocí vzorce 20 převedeny na invariantní momenty.

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{\left(1+\frac{i+j}{2}\right)}} \tag{20}$$

Posléze bylo využito následujících sedmi rotačně invariantních momentů:

$$\begin{aligned}
I_1 &= \eta_{2,0} + \eta_{0,2} \\
I_2 &= (\eta_{2,0} - \eta_{0,2})^2 + 4\eta_{1,1}^2 \\
I_3 &= (\eta_{3,0} - 3\eta_{1,2})^2 + (3\eta_{2,1} - \eta_{0,3})^2 \\
I_4 &= (\eta_{3,0} + \eta_{1,2})^2 + (\eta_{2,1} + \eta_{0,3})^2 \\
I_5 &= (\eta_{3,0} - 3\eta_{1,2})(\eta_{3,0} + \eta_{1,2})\left[(\eta_{3,0} + \eta_{1,2})^2 - 3(\eta_{2,1} + \eta_{0,3})^2\right] + \\
&\quad (3\eta_{2,1} - \eta_{0,3})(\eta_{2,1} + \eta_{0,3})\left[3(\eta_{3,0} + \eta_{1,2})^2 - (\eta_{2,1} + \eta_{0,3})^2\right] \\
I_6 &= (\eta_{2,0} - \eta_{0,2})\left[(\eta_{3,0} + \eta_{1,2})^2 - (\eta_{2,1} + \eta_{0,3})^2\right] + 4\eta_{1,1}(\eta_{3,0} + \eta_{1,2})(\eta_{2,1} + \eta_{0,3}) \\
I_7 &= (3\eta_{2,1} - \eta_{0,3})(\eta_{2,1} + \eta_{0,3})\left[3(\eta_{3,0} + \eta_{1,2})^2 - (\eta_{2,1} + \eta_{0,3})^2\right] - \\
&\quad (\eta_{3,0} - 3\eta_{1,2})(\eta_{2,1} + \eta_{0,3})\left[3(\eta_{3,0} + \eta_{1,2})^2 - (\eta_{2,1} + \eta_{0,3})^2\right]
\end{aligned} \tag{21}$$

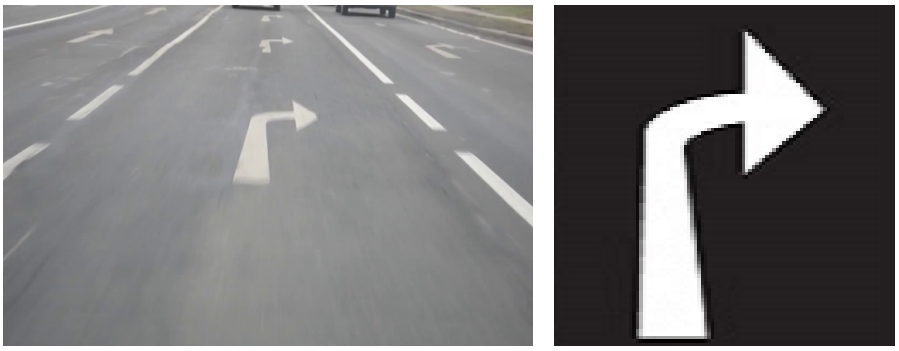
Výhodou použití těchto rotačně invariantních (neměnných) momentů je, že nejsou ovlivněny posunutím, změnou měřítka nebo rotací. Např. moment I_7 je invariantní vůči zkosení, což umožňuje rozlišit zrcadlové obrazy jinak stejných obrázků. Těchto sedm vypočtených momentů se vložilo do vektoru příznaků. Využití tohoto vektoru příznaků je popsáno v další části této diplomové práce.

3.7 Porovnání vlastností a určení dopravní značky

Předposledním krokem, který se provádí při detekci dopravního značení, je porovnání příznaků (vlastností) daného největšího objektu s předem vypočtenými hodnotami stanovujícími a charakterizujícími jednotlivé typy šipek. Nejdříve je nutné vypočítat euklidovskou vzdálenost vektoru objektu k vektoru každé šipky. Tyto hodnoty vzdáleností aplikace posléze seřadí od nejmenší po největší. Následně se porovná s dalšími příznaky charakterizujícími danou šipku. Mezi porovnávané vlastnosti patří pozice těžiště objektu vzhledem ke středu této vyříznuté oblasti, dále pravoúhlost objektu, podlouhlost objektu, kruhovost objektu, poměr obsahu objektu a obsahu konvexního obalu objektu. U některých šipek se využívá i úhlu, pod kterým se opsal nejmenší pravoúhelník nebo polohy horního extrému dané šipky (šipka rovně a doprava ho má mít v levé části a šipka rovně a doleva pak v pravé části). Tyto vlastnosti byly nejprve vypočteny na základě testovacích dat pro všechny dopravní značky. Posléze se z těchto dat vytvořily intervaly charakterizující jednotlivé dopravní značky.

Pokud tyto vlastnosti u detekovaného objektu patří do intervalů, které jsou charakteristické pro danou dopravní značku, pak lze tento objekt za tuto dopravní značku považovat.

Poslední podmínkou pro vykreslení dopravní značky je, že do okamžiku opětovné detekce vozovky (v dané prohledávané oblasti se nenachází žádný dostatečně velký objekt) byla tato dopravní značka detekována alespoň dvakrát. To se provádí za účelem snížení možné chybné detekce. Jak vypadá úspěšná detekce je patrné z obrázku 20.



(a) (b)

Obrázek 20: Výsledek po úspěšném rozpoznání dopravní značky

3.8 Detekce dopravních čar

Mimo detekce dopravních značek se tato diplomová práce zabývá i detekcí dopravního pruhu, ve kterém se vozidlo nachází. Aplikace do videozáznamu vyznačí podélné středové a okrajové čáry označující příslušný jízdní pruh.

Vstupní obraz je nejprve, z důvodů požadavků dalšího zpracování, převeden na obraz ve stupních šedi. V dalším kroku se na obraz provede rozmazání pomocí Gaussovy metody. Takto upravený obraz je připraven pro následnou detekcí hran.

3.8.1 Detekce hran

V aplikaci se pro detekci hran využívá gradientní metoda. Tato metoda využívá pro výpočet hodnoty jasu vždy dvou sousedních bodů a to jak ve směru x, tak ve směru y.

$$\partial x = \frac{f\left(x+1, y\right) - f\left(x-1, y\right)}{2} \tag{22}$$

$$\partial y = \frac{f\left(x, y+1\right) - f\left(x, y-1\right)}{2} \tag{23}$$

$$\partial e(x, y) = \sqrt{\partial x^2 + \partial y^2} \tag{24}$$

Z použitého vzorce je patrné, že gradient se vypočítává pro každý jednotlivý bod obrazu. Proto byla tato metoda (z důvodu zrychlení zpracování) napsána tak, aby zpracování jednotlivých bodů probíhalo paralelně, přičemž se využilo technologie CUDA na GPU.

Obrázek 21 ukazuje, jak vypadá detekce hran když se bod, který leží na hranici objektu, vykreslí v hodnotě jasu, která odpovídá vypočtené hodnotě ze vzorce 24. V případě použité metody se následně provede prahování a každá vypočtená hodnota pixelu, která je větší než předem stanovený práh, se přepíše na hodnotu 255. V opačném případě se přepíše na hodnotu 0.

Jak vyplývá z následující ukázky kódu 1, dochází k tomu, že každé vlákno zpracovává výpočet hodnoty gradientu pro jeden pixel. V případě, že daný bod je součástí nějaké hrany, pak je označen bílou barvou. Pokud tomu tak není, je označen barvou černou. Po provedení této operace získáme binární obraz.

(a) (b)

Obrázek 21: Vizualizace detekce hran v obraze

```
__global__ void kernel_CenterDifference( Matrix input, Matrix output)
{
    int row = blockDim.y * blockIdx.y + threadIdx.y;
    int col = blockDim.x * blockIdx.x + threadIdx.x;
    if ( col >= input.width ) return;
    if ( row >= input.height ) return;

    double derivateX, derivateY, edgeDerivate;
    if ((col > 5 && col < input.width-5) && (row > 5 && row < input.height-5)){
        derivateX = (input.elements[row * input.width + col + 1] -
                    input.elements[row * input.width + col - 1])/2;
        derivateY = (input.elements[(row+1) * input.width + col] -
                    input.elements[(row-1) * input.width + col]) /2;
        edgeDerivate = Edge(derivateX,derivateY);
        if (edgeDerivate >=10){
            SetElement(output,row, col, 255);
        }
        else{
            SetElement(output,row, col, 0);
        }
    }
    else{
        SetElement(output,row, col, 0);
    }
}
```

Výpis 1: Metoda pro detekci hran prováděná na GPU

3.8.2 Houghova transformace

Jakmile je získán binární obraz, je na tento obraz aplikována metoda Houghovy transformace. Pomocí ní zjistíme všechny body, které leží na nějaké přímce nebo úsečce přičemž poloha této přímky nám není předem známá. Podrobněji je tato metoda popsána v kapitole 2.9. Stejně jako v předchozí metodě pro nalezení hran (i tady dochází ke zpracování jednotlivých pixelů obrazu), byla tato metoda napsána s využitím technologie CUDA na GPU. V metodě se pro každý úhel v intervalu $0°$ až $180°$ tento úhel nejprve převede na radiány.

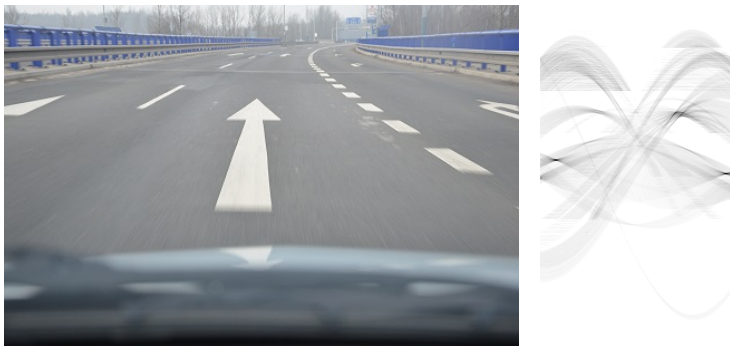
$$a = \frac{\alpha}{180}\pi \tag{25}$$

Následně se vypočte hodnota ρ

$$\rho = x_i \cos(a) + y_j \sin(a), \tag{26}$$

kde a označuje úhel v radiánech, α úhel ve stupních a ρ kolmou vzdálenost pixelu od počátku souřadného systému (levý horní bod na souřadících (0,0)).

Poté inkrementujeme hodnotu v akumulátoru na vypočtené pozici. Akumulátor je pole o rozměrech 180 (maximální úhel) na šířku a $\sqrt{width^2 + height^2}$, kde $width$ a $height$ je šířka respektive výška vstupního obrazu, pro výšku pole. Akumulátor o těchto rozměrech pokrývá celý vstupní obraz (maximální vzdálenost ρ je uhlopříčka vstupního obrazu). Jak vypadá toto pole po vizuální stránce je ukázáno na obrázku 22(b).



(a) (b)

Obrázek 22: Vizualizace hodnot z akumulátoru

Obrázek 22(b) nám ukazuje kolik bodů n_{ij} leží na přímce s parametry (θ_i, ρ_j). Čím je daný bod v našem případě tmavší, tím více bodů na dané přímce leží. Pro odstranění nežádoucích krátkých přímek stačí vybírat jen ty přímky, na kterých leží více bodů než je nějaká předem definovaná prahová hodnota. V případě této aplikace je tato prahová hodnota nastavena na 60 což znamená, že přímka musí obsahovat alespoň 60 bodů.

```
__global__ void kernel_HoughTransform( Matrix input, Matrix output, double center_x,
 double center_y, double hough_h){
   int row = blockDim.y * blockIdx.y + threadIdx.y;
   int col = blockDim.x * blockIdx.x + threadIdx.x;
   if ( col >= input.width ) return;
   if ( row >= input.height ) return;

   output.elements[row * input.width + col] = 0;
   int w = input.width;
   double rho;
   int thetaDeg;

   if (input.elements[row * w + col] > 0){
      // Kde cos_Theta_array a sin_Theta_array jsou pole
      // obsahujici predvypocitane hodnoty (sin(theta),cos(theta))
      for(thetaDeg = 40; thetaDeg <= 60 ; thetaDeg++)
      {
         rho = (((double)col - center_x) * cos_Theta_array[thetaDeg]) +
           (((double) row - center_y) * sin_Theta_array[thetaDeg]);
         output.elements[(int)(Round(rho + hough_h) * 180.0) + thetaDeg]++;
      }
      for(thetaDeg = 120; thetaDeg <= 140 ; thetaDeg++)
      {
         rho = (((double)col - center_x) * cos_Theta_array[thetaDeg]) +
           (((double) row - center_y) * sin_Theta_array[thetaDeg]);
         output.elements[(int)(Round(rho + hough_h) * 180.0) + thetaDeg]++;
      }
   }
}
```

Výpis 2: Houghova transformace prováděná na GPU

Poté, co jsme v obraze nalezli všechny přímky, je potřeba zjistit krajní body každé z těchto přímek. I v tomto případě lze zpracovávat každý bod z akumulátoru (tedy každou přímku) samostatně. Zpracování přímek jde zrychlit pomocí paralelního zpracování na grafické kartě a to za využití technologie CUDA.

Za účelem výběru požadovaných přímek je potřeba využít určitých kritérií. Jak bylo popsáno výše, jedním z kritérií bude počet bodů ležících na dané přímce. Druhým kritériem, kterého se dá využít, je výběr na základě úhlů, pod kterým daná přímka vede. Je možné předpokládat, že lze vyloučit všechny přímky, které jsou vodorovné. Na základě testovaní různých kritérií pro výběr přímek pod určitým úhlem (označení θ) byla zvolena následující kritéria:

$$\theta \geq 40° \text{ a zároveň } \theta \leq 60° \text{ nebo } \theta \geq 120° \text{ a zároveň } \theta \leq 140°$$

První část podmínky vybere přímky, které by měly popisovat vyznačení pruhu z levé strany a druhá část přímky popisující vyznačení pruhu z pravé strany. Dalším kritériem je vybrání jen těch přímek, které z podobných (případně vodorovných) přímek v blízkém okolí obsahují nejvíce bodů. Tedy pro každou přímku uloženou v akumulátoru se pomocí masky o rozměrech 21×21, kde daná přímka leží uprostřed této masky, zjistí zda je v tomto okolí maximem.

```
__global__ void kernel_GetLines(Matrix hough_input, int *Ax, int *Bx,
int *Ay, int*By, double img_width, int img_height)
{
    int rho = blockDim.y * blockIdx.y + threadIdx.y;
    int theta = blockDim.x * blockIdx.x + threadIdx.x;
    if ( theta >= hough_input.width ) return;
    if ( rho   >= hough_input.height ) return;

    int w = hough_input.width;
    int h = hough_input.height;
    int threshold = 60;
    int index = theta + (rho * w);
    Ax[index] = Ay[index] = -10;
    Bx[index] = By[index] = -10;

    if (hough_input.elements[theta + (rho * w)] >= threshold){
        int max = hough_input.elements[theta + (rho * w)];
        for(int ly=-10;ly<=10;ly++){
            for(int lx=-10;lx<=10;lx++){
                if ( (ly + rho >= 0 && ly + rho < h) && (lx + theta >= 0 && lx + theta < w) ){
                    if ( (int)hough_input.elements[( (rho+ly)*w) + (theta+lx)] > max ){
                        max = hough_input.elements[( (rho+ly)*w) + (theta+lx)];
    }   }   }   }
        if (max == (int)hough_input.elements[(rho*w) + theta])
        {
            //Stredova cara
```

```
            if (theta >= 40 && theta <= 60)
            {
                //y = (r − x cos(t)) / sin(t); x = (r − y sin(t)) / cos(t);
                //sin_Theta_array a cos_Theta_array jsou predvypocitane hodnoty
                Ay[index] = 0;
                Ax[index] = ((double)(rho − (h/2)) − ((Ay[index] − (img_height/2)) ∗
                sin_Theta_array[theta])) / cos_Theta_array[theta] + (img_width/2);
                Bx[index] = 0;
                By[index] = ((double)(rho − (h/2)) − ((Bx[index] − (img_width/2) ) ∗
                cos_Theta_array[theta])) / sin_Theta_array[theta] + (img_height/2);
            }
            //Okrajova cara
            else if (theta >= 120 && theta <= 140)
            {
                Ay[index] = 0;
                Ax[index] = ((double)(rho − (h/2)) − ((Ay[index] − (img_height/2)) ∗
                sin_Theta_array[theta])) / cos_Theta_array[theta] + (img_width/2);
                Bx[index] = img_width;
                By[index] = ((double)(rho − (h/2)) − ((Bx[index] − (img_width/2) ) ∗
                cos_Theta_array[theta])) / sin_Theta_array[theta] + (img_height/2);
            }
        }
    }
}
```

Výpis 3: Nalezení krajních bodu přímky

Pokud jsou splněny všechny předchozí podmínky dojde k výpočtu souřadnic krajních bodů, pomocí kterých se tato přímka vykreslí. Souřadnice těchto krajních bodů byly vypočteny vždy s jednou známou hodnotou a to tak, aby daný bod měl souřadnici x rovnu 0 nebo šířce vstupního obrazu. Druhá souřadnice, tedy y, se vypočetla na základě souřadnice x podle následujícího vzorce:

$$y = \frac{\rho - x \cos a}{\sin a}, \tag{27}$$

kde a označuje úhel v radiánech a ρ kolmou vzdálenost pixelu od počátku souřadného systému (levý horní bod na souřadících (0,0)).

3.9 Vyznačení čar v obraze

V okamžiku, kdy jsou nalezeny všechny přímky které splňují dříve popsaná kritéria, čímž jsme získali krajní body těchto přímek, je potřeba tyto přímky vykreslit do obrazu. Z důvodu že jedinými známými body přímek jsou body krajní, je potřeba zbylé body dopočítat.

Nejprve je nezbytné zjistit maximální počet bodů ležících mezi těmito krajními body. Jestliže krajní body přímky označíme jako body A a B, kde daný bod leží na souřadnicích x a y, pak tento počet bodů je maximálně roven větší z hodnot vzdálenosti souřadnic přímky tedy rozdílu mezi B_x a A_x respektive B_y a A_y. Pak platí následující vzorec:

$$\text{max\_distance} = \begin{cases} \mid B_x - A_x \mid & \text{pro } \mid B_x - A_x \mid \geq \mid B_y - A_y \mid, \\ \mid B_y - A_y \mid & \text{pro } \mid B_x - A_x \mid < \mid B_y - A_y \mid \end{cases} \tag{28}$$

Poté, co jsme zjistili počet bodů které je potřeba vykreslit, přejdeme k výpočtu souřadnic jednotlivých bodů. Bude se vycházet ze vzorce 31 pro parametrického vyjádření přímky. Nejprve se vypočte směrový vektor přímky $u\,(u_1, u_2)$.

$$\begin{aligned} u_1 &= B_x - A_x \\ u_2 &= B_y - A_y \end{aligned} \tag{29}$$

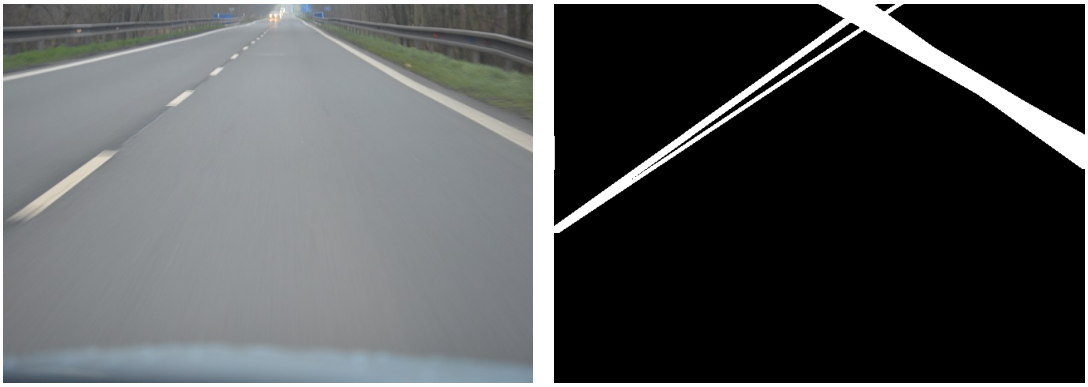
Po vypočtení směrového vektoru dojde k výpočtu jednotlivých bodů. Tento výpočet se bude provádět v cyklu pro všechny t, kdy na začátku cyklu bude $t = 1$ a v každé iteraci se t vypočte z předchozí hodnoty na základě následujícího vzorce.

$$t_k = t_{k-1} - \frac{1}{\mid \text{max\_distance} \mid} \tag{30}$$

Samotný výpočet souřadnic je dán vzorcem:

$$\begin{aligned} X &= A_x + tu_1 \\ Y &= A_y + tu_2 \end{aligned} \tag{31}$$

Takto vypočtený bod se do výsledného obrazu, který na vstupu do metody obsahuje pouze černou barvu (hodnotu 0), zakreslí bílou barvou, tedy zapíše se na danou pozici x a y hodnota 255. Jak vypadá ukázka tohoto vykreslení je ukázáno na obrázku 23.



(a) (b)

Obrázek 23: Vykreslení přímek na základě souřadnic krajních bodů

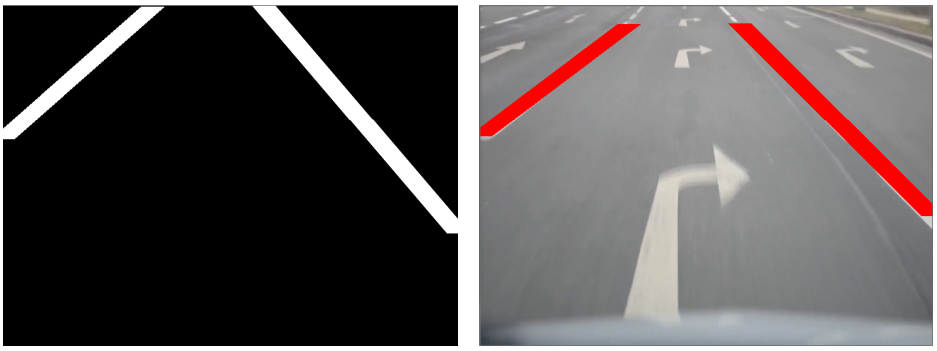
I v tomto případě lze využít paralelní zpracování a vykreslovat více přímek současně.

```
__global__ void kernel_drawline( Matrix Lines, int *Ax , int *Ay, int *Bx, int *By,
  int L, int thickness)
{
   int l = blockDim.x * blockIdx.x + threadIdx.x;
   if ( l >= L ) return;

   int w = Lines.width;
   int h = Lines.height;

   if ((Ax[l] == -10) && (Ay[l] == -10) && (Bx[l] == -10) && (By[l] == -10))
   {  return;  }
   else if ((By[l] > 0 && By[l] < h) && (Ax[l] > w*0.2 && Ax[l] < w*0.8)){
      int X,Y, int_x, int_y;
      double t = 1.0, repeat_num;
      int direction_x = 0, direction_y = 0;

      direction_x = Bx[l] - Ax[l];
      direction_y = By[l] - Ay[l];
      repeat_num = (abs(direction_x) >= abs(direction_y))?direction_x:direction_y;

      while( t > 0)
      {
         int_x = (Ax[l] + t * direction_x);
         int_y = (Ay[l] + t * direction_y);
         X = (((Ax[l] + t * direction_x) - int_x) >= 0.5) ? int_x +1 : int_x ;
         Y = (((Ay[l] + t * direction_y) - int_y) >= 0.5) ? int_y +1 : int_y ;
         for(int s = -(thickness); s <= (thickness); s++)
```

```
                {
                    if (((X + s >= 0) && (X + s < Lines.width)) && (Y >= 0) && (Y < Lines.height))
                    {
                        Lines.elements[Y * Lines.width + X + s ] = 255;
                    }
                }
                t = t - 1/abs(repeat_num);
            }
        }
    }
```

Výpis 4: Vykreslení čar do obrazu

Matice, která se získala vykreslením přímek vyznačujících podélné středové a okrajové čáry, se použije jako šablona pro konečné vyznačení pruhu do původního barevného snímku z videozáznamu. Vychází se z jednoduchého principu. Je-li v matici, získané v předchozím kroku, na dané pozici hodnota 255, do barevného obrazu se daný bod vykresli červeně. Pokud je hodnota rovná 0, pak se zachová původní hodnota z barevného obrazu. Výsledek tohoto vykreslení je znázorněn na obrázku 24.



(a) (b)

Obrázek 24: Vykreslení detekovaných čar do barevného obrazu

Uvedenou operací program na detekci vodorovného dopravního značení a dopravních pruhů ukončí zpracování jednoho snímku. Poté se celý proces opakuje pro každý další snímek ze zdrojového videa.

4 Zhodnocení dosažených výsledků

V kapitole 4 jsou uvedeny dosažené výsledky, jenž byly získány během testování aplikace. Jsou zde popsány problémy chybných detekcí vodorovného dopravního značení.

Testovaní aplikace bylo prováděno na videozáznamech pořízených z jedoucího automobilu v různých částech města Ostravy. Jednalo se o městské obvody Poruba, Zábřeh, Marianské Hory a Hulváky, Slezská Ostrava a Výškovice. Videa byla pořízena pomocí kamery Panasonic HDC-SD10 a digitální zrcadlovky Canon. Automobil se po vozovce pohyboval rychlostí v rozmezí 40 až 90 km/h. Pořizování videozáznamů probíhalo převážně během dne. Aplikace byla testována i na videozáznamu v noci. V případě nočního videa je nutno upravit prahovací hodnotu a to z důvodu zlepšení detekce objektu. Celková délka videozáznamů, na kterých byla aplikace testována, je 23 minut a 25 sekund. V těchto videozáznamech se nacházelo celkem 194 dopravních značek, které je aplikace schopná rozpoznat.

V následující tabulce jsou uvedeny dosažené výsledky získané během testování.

Počet značek	**Správně určených**	**Neurčených**	**Chybně určených**	**Správně určených v %**
194	176	17	1	90.7

Tabulka 1: Vyhodnocení detekce dopravních značek

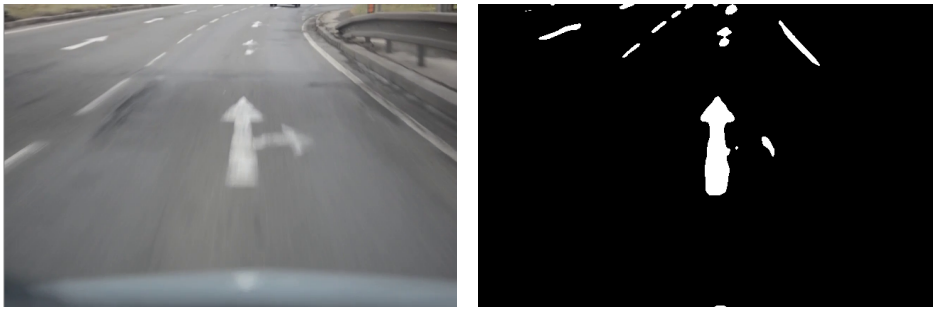
Na základě údajů z tabulky 1 můžeme říci, že aplikace dokázala správně rozpoznat 90.7% vodorovných dopravních značek. Tento výsledek správného určení vodorovné dopravní značky lze brát jako velmi dobrý. Dopravní značky, které aplikace nebyla schopna určit, byly detekovány pouze jednou. Tím nebyla splněna podmínka dvou a více úspěšných rozpoznání.

Tabulka 2 udává množství jednotlivých dopravních značek a u každé této značky je popsáná procentuální správnost jejich určení.

Název	Počet značek	Správně určených	Neurčených	Chybně určených	Spravně určených v %
Rovně	59	54	5	0	91.5%
Rovně a doleva	21	18	3	0	86%
Rovně a doprava	56	48	7	1	85.7%
Doleva	21	19	2	0	90.5%
Doprava	26	26	0	0	100%
Spojení doleva	7	7	0	0	100%
Spojení doprava	4	4	0	0	100%

Tabulka 2: Výskyt a detekce jednotlivých dopravních značek

Z tabulky 2 vyplývá, že nejčastější výskyt má na pořízených videozáznamech dopravní značka: "šipka rovně", "šipka rovně a doprava". Ve většině případů, kdy nedošlo ke správnému rozpoznání dopravní značky, byla dopravní značka na videozáznamu buď ve špatném stavu, nebo to bylo např. při průjezdu pod mostem, kdy došlo k výrazné změně jasu. V jednom případě byla dopravní značka detekovaná špatně, bylo to způsobeno tím, že po prahování byla dopravní značka více podobná šipce rovně než šipce rovně a doprava. Tento případ je znázorněn na obrázku 25.



(a) Původní snímek z videozáznamu (b) Obraz získaný po prahování

Obrázek 25: Ukázka nerozpoznání dopravní značky

Detekce podélných středových a okrajových čar splňovala zadané požadavky. Problém s detekcí nastal v okamžiku, kdy se vozidlo nacházelo v zatáčce. Důvodem bylo skutečnost, že aplikace vyhledávala čáry pod určitým úhlem tj. 40 až 60 respektive 120 až 140 stupňů. Toto kriterium v zatáčkách nebylo možné splnit a tedy podélné čáry správně detekovat.

Aplikace byla testována na počítači s konfigurací, která je uvedena v následující tabulce. Bylo využito knihoven OpenCV ve verzi 2.4.9 a CUDA 5.5.

	Konfigurace
Procesor	Intel Core i3 2310M (2,1 GHz)
Paměť RAM	4 GB
Grafická karta	NVIDIA GeForce 410M, 48 CUDA Cores
Operační systém	MS Windows 7

Tabulka 3: Konfigurace počítačů, na kterých probíhalo testování

Průměrná časová náročnost operací prováděných na každém snímku byla cca 17 ms. Tento čas byl v průměru z 90% tvořen detekcí čar. Rozpoznávání dopravních značek mělo průměrný čas do 3 ms. V případě detekce čar zabírá většinu času přenášení dat z grafické karty na procesor.

4.1 Porovnání s bakalářskou prací

Detekce vodorovného dopravního značení byla v rámci mé bakalářské práce řešena pomocí metody MatchTemplate. Byla použita šablona obsahující všechny dopravní značky, které byla aplikace schopna rozpoznat. Obrázek detekované dopravní značky z vozovky byl následně porovnán s touto šablonou. Pokud došlo ke shodě mezi detekovanou dopravní značkou a některou z dopravních značek na šabloně, bylo toto aplikací znázorněno v samostatném okně.

V diplomové práci bylo pro rozpoznávání vodorovných dopravních značek využito příznaků a vlastností objektů. Mezi tyto vlastnosti patřily např. pravoúhlost, podlouhlost, kruhovost. Tyto vlastnosti byly vypočteny pro všechny dopravní značky z trénovací množiny. Následně byly vytvořeny intervaly charakterizující jednotlivé dopravní značky. Pokud vlastnosti detekované dopravní značky patřily do intervalů charakterizujících příslušnou dopravní značku, pak byla za tuto dopravní značku určena. Diplomová práce obsahuje navíc detekci pruhů. Při detekci pruhů bylo využito GPU a platformy CUDA. Pro rozpoznávaní vodorovných dopravních značek, kde se průměrně čas pohyboval pod 3 ms nebylo nutné využít paralelního zpracování.

Výsledky obou použitých způsobů jsou srovnatelné. Vyjímku tvoří to, že v bakalářské práci nastane problém s rozpoznáváním vodorovné dopravní značky v případě, kdy detekovaná dopravní značka je zkosená nebo natočená.

Úspěšnost detekce vodorovného dopravního značení v rámci bakalářské práce je uvedena v následující tabulce.

Počet značek	**Správně určených**	**Neurčených**	**Chybně určených**	**Spravně určených v %**
162	146	12	4	90%

Tabulka 4: Vyhodnocení počtu správných určení

V bakalářské práci se čas zpracování jednoho snímku pohyboval okolo 60 ms na počítači s obdobnou konfigurací. V rámci mé bakalářské práce nebyla provedena detekce podélných středových a okrajových čar.

4.2 Porovnání CPU a GPU verze

Aplikace byla testována ve třech verzích detekce podélných středových a okrajových čar. Jednalo se o verze GPU, CPU a CPU paralelně. V případě CPU paralelní verze bylo využito knihovny OpenMP [12]. Knihovna OpenMP umožnila při zpracování využít 4 paralelních vláken procesoru na testovacím počítači (2 jádrový procesor). Časy vybraných operací jsou uvedeny v tabulce 5 a 6.

Operace	**GPU**	**CPU**	**CPU paralelně**
Detekce hran	0.66 ms	13.5 ms	10.5 ms
Houghova transformace	4 ms	39 ms	8 ms
Detekce čar	**14.5 ms**	**75.5 ms**	**38.5 ms**

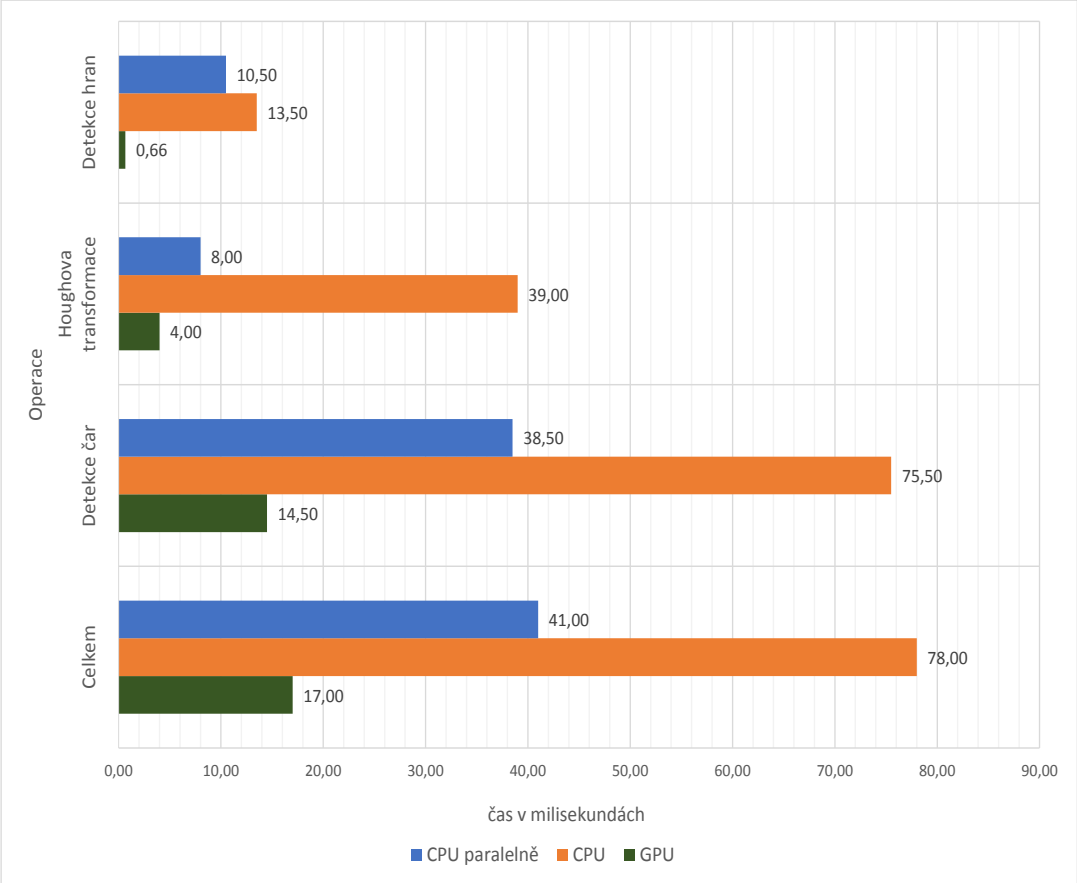
Tabulka 5: Porovnaní CPU a GPU verze u vybraných operací detekce čar

Operace	**GPU**	**CPU**	**CPU paralelně**
Detekce čar	14.5 ms	75.5 ms	38.5 ms
Rozpoznávaní značek	2.5 ms	2.5 ms	2.5 ms
Celkově	**17 ms**	**78 ms**	**41 ms**

Tabulka 6: Porovnaní CPU a GPU verze výsledných časů zpracovaní snímku

Z tabulky je patrné, že nejnižších časů je dosaženo ve verzi GPU. Velké rozdíly v čase jsou i mezi jednotlivých verzemi na CPU. Z tabulky také vyplývá, že časově nejnáročnější je operace Houghovy transformace.

Pro lepší porovnání časů byl použit následující pruhový graf.



Obrázek 26: Porovnání časů zpracování jednotlivých operací

5 Závěr

Tématem mé diplomové práce byla detekce vodorovného dopravního značení s využitím GPU. Cílem bylo vytvořit algoritmus pro detekci vodorovného dopravního značení na základě analýzy obrazů získaných kamerou umístěnou v automobilu.

Diplomovou práci jsem rozdělil do dvou částí. V teoretické části jsem se zaměřil na popsaní metod, které byly následně využity v rámci vlastní implementace. Pro lepší názornost jsem použil u některých metod i grafické znázornění.

Druhou část mé diplomové práce tvoří vlastní implementace řešení. Nejprve jsem musel pořídit videozáznamy vodorovného dopravního značení. Videa jsem pořizoval v rámci města Ostravy v průběhů měsíců září loňského roku a letošního ledna. Celková délka pořízeného videozáznamu, který byl použit pro testovaní aplikace, je 23 minut a 25 sekund. Pořízená videa, z důvodu snížení velikosti, jsem zmenšil na rozlišení 640×480. Takto upravený videozáznam jsem následně zpracoval pomocí aplikace na detekci vodorovného dopravního značení. Tato aplikace řeší dva úkoly současně. Jednak detekuje podélné středové a okrajové čáry a zároveň s tím detekuje v obraze vodorovné dopravní značky, které leží v dopravním pruhu, ve kterém se vozidlo s videozáznamem pohybuje.

Výsledky rozpoznávání vodorovného dopravního značení jsou popsány v kapitole Zhodnocení dosažených výsledků. Z pořízených videozáznamů je aplikace schopna rozpoznat 176 z celkového počtu 194 dopravních značek. Vyčísleno v procentech je úspěšnost rozpoznání 90.7%. Tento výsledek považuji za velmi dobrý. Případy, kdy nedošlo ke správnému rozpoznání dopravní značky byla tato dopravní značka na videozáznamu ve špatném stavu nebo se jednalo o situaci, kdy automobil projížděl pod mostem a tedy došlo k výrazné změně jasu. Pouze v jednom případě byla dopravní značka detekována špatně a to z důvodu, že po prahování byla více podobná šipce rovně než šipce rovně a doprava.

Mnou zpracovaná aplikace pro detekci podélných středových a okrajových čar a směrových šipek, by se v budoucnu mohla rozšířit o detekci vozidel jedoucích před námi a dodržování bezpečné vzdálenosti mezi jedoucími vozidly. V úvahu připadá i detekce překážek na vozovce, popřípadě detekce chodců.

Bc. Jan Štěpán

6 Reference

[1] BRADSKI, Gary R. *Learning OpenCV.* Sebastopol: O'Reilly, 2008, 555 s. ISBN 978-0-596-51613-0.

[2] DOBEŠ, Michal. *Zpracování obrazu a algoritmy v C#.* 1. vyd. Praha: BEN - technická literatura, 2008, 143 s. ISBN 978-80-7300-233-6.

[3] FIŘT, Jaroslav a Radek HOLOTA. *Digitalizace a zpracování obrazu*, 5 s. Dostupné z: http://home.zcu.cz/~holota5/publ/DigZprO.pdf.

[4] *Gimp: Documentation* [online]. [cit. 2015-04-04]. Dostupné z: http://docs.gimp.org/2.2/cs/plug-in-convmatrix.html.

[5] HLAVÁČ, Václav. *Jasové a geometrické transformace.* [online prezentace]. Praha : České Vysoké Učení Technické v Praze, 2009,[cit. 2015-04-04]. Dostupný z WWW: http://cmp.felk.cvut.cz/~hlavac/TeachPresCz/11DigZprObr/18BrightGeomTxCz.

[6] HLAVÁČ, Václav a Milan ŠONKA. *Počítačové vidění*. 1. vyd. Praha: Grada, 1992, 272 s. ISBN 80-854-2467-3.

[7] HSL and HSV: Basic principle. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-02-20]. Dostupné z: http://en.wikipedia.org/wiki/HSL_and_HSV

[8] KUČERA, Jiří. *Shluková analýza: Algoritmus k-means*. [online]. [cit. 2015-04-19]. Dostupné z: http://is.muni.cz/th/172767/fi_b/5739129/web/web/kmeans.html

[9] KHEYROLLAHI, Alireza a Toby P. BRECKON. Automatic real-time road marking recognition using a feature driven approach. *Machine Vision and Applications*, 2012, 1-11. [online]. 2012, roč. 23, č. 1, s. 123-133 [cit. 2015-03-04]. ISSN 0932-8092. DOI: 10.1007/s00138-010-0289-5. Dostupné z: http://link.springer.com/10.1007/s00138-010-0289-5.

[10] *Ministerstvo dopravy* [online]. 2006 [cit. 2015-03-25]. Dostupné z: http://www.mdcr.cz/cs/default.htm

[11] *OpenCV: Documentation* [online]. 2010, 2013 [cit. 2015-04-04]. Dostupné z: http://docs.opencv.org/index.html.

[12] *OpenMP and C++: Reap the Benefits of Multithreading without All the Work.* GATLIN, Kang Su a Pete ISENSEE. [online]. [cit. 2015-04-28]. Dostupné z: https://msdn.microsoft.com/en-us/magazine/cc163717.aspx

[13] POLICIE ČESKÉ REPUBLIKY. *Statistika nehodovosti* [online]. 2014. vyd. 2014 [cit. 2015-03-08]. Dostupné z: http://www.policie.cz/clanek/statistika-nehodovosti-900835.aspx

[14] RGB color model: Additive primary colors. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-04-04]. Dostupné z: http://en.wikipedia.org/wiki/RGB_color_model

[15] SHAPIRO LINDA, G. *Computer Vision*. Vyd. 1. New Jersey: Prentice-Hall, 2001, 580 s. ISBN 01-303-0796-3.

[16] SOCHOR, Jakub. *Rychlá detekce dopravních značek v obraze.* Brno, 2012. 51 s. Bakalářská práce. Vysoké Učení Technické v Brně. Vedoucí práce Doc. Ing. Adam Herout, Ph.D.

[17] SOJKA, Eduard. *Digitální zpracování a analýza obrazů.* 1. vyd. Ostrava: VŠB - Technická univerzita, 2000, 133 s. ISBN 80-707-8746-5.

[18] SURYNKOVÁ, Petra. *Počítačová geometrie: Konvexní obal a množina* [online prezentace]. Praha: Fakulta matematiky a fyziky, KU, [cit. 2015-03-08]. Dostupné z: http://surynkova.info/dokumenty/mff/PG/Prednasky/prednaska_8.pdf

[19] ŠTĚPÁN, Jan. *Detekce vodorovného dopravního značení v obrazech*: bakalářská práce. Ostrava: VŠB - Technická univerzita, Fakulta elektrotechniky a informatiky, 2013, 34 s. Vedoucí práce byl Ing. Michael Holuša.

[20] Zákon č.361/2000 Sb., o provozu na pozemních komunikacích , § 62 a § 64.

[21] ŽÁRA, Jiří, Bedřich BENEŠ, Jiří SOCHOR a Petr FELKEL. *Moderní počítačová grafika*. Vyd 1. Brno: Computer Press, 2004, 609 s. ISBN 8025104540.